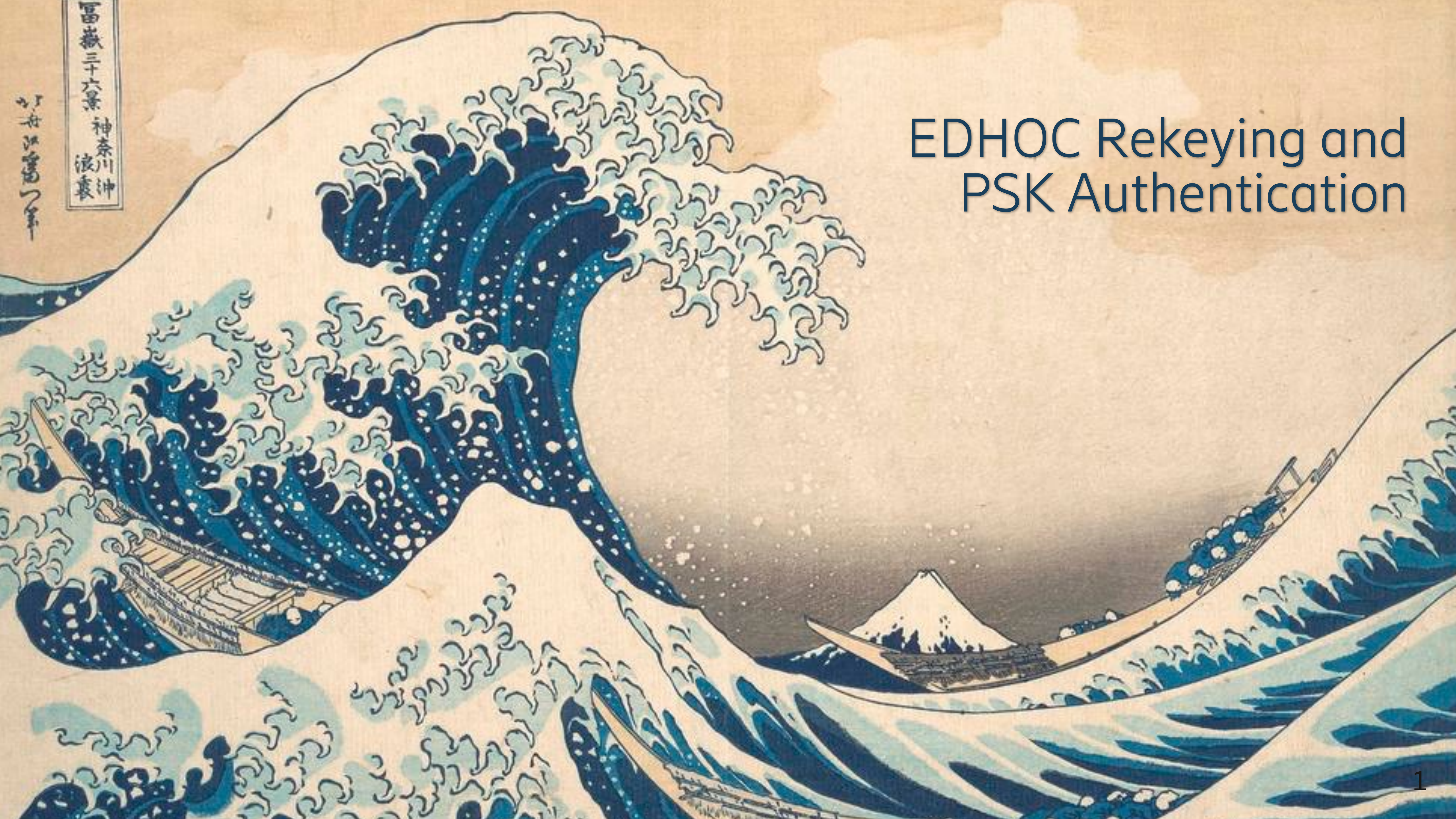


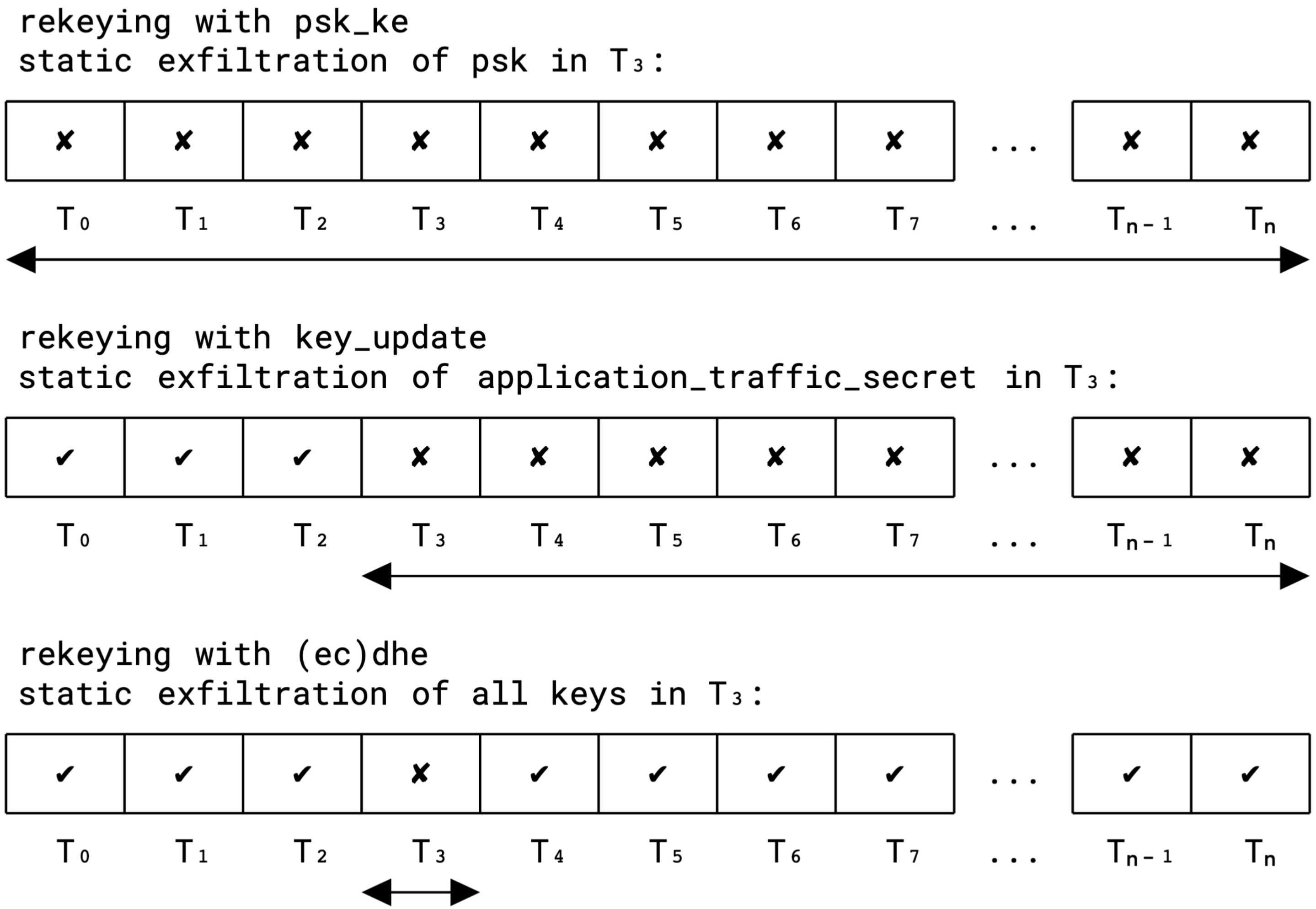
EDHOC Rekeying and PSK Authentication

EDHOC rekeying and PSK authentication

- PSK authentication and resumption were discussed when LAKE was formed but not pursued.
- **New motivation:** Rafael Marín López asked for a lightweight resumption mechanism.
- A flexible way to achieve this is to standardize a new PSK authentication method that can be used for resumption similar to the PSK/resumption mechanism in TLS 1.3. This can then also be used with an external PSK.
- TLS will likely discourage use of the the psk_ke method without forward secrecy. Reuse of tickets, external psk identifiers, or key shares enables tracking, fingerprinting, and identification of the client and server. Harder requirements for reuse of tickets, external PSK identifiers, and key shares have recently been merged to RFC8446bis.
  - Appendix C.4 of https://tlswg.org/tls13-spec/draft-ietf-tls-rfc8446bis.txt
  - https://datatracker.ietf.org/doc/draft-mattsson-tls-psk-ke-dont-dont-dont/
- Any new EDHOC method should provide ephemeral key exchange, identity protection, and mitigate tracking and fingerprinting. Anonymous identifiers that are only used once provide identity protection and mitigates tracking. Due to synchronization issues, we might have to accept something slightly worse than that.

Key exchange without forward secrecy enables passive monitoring

- Key exchange without forward secrecy enables passive monitoring.
- Malicious actors can get access to long-term keys in different ways: physical attacks, hacking, social engineering attacks, espionage, or by simply demanding access to keying material with or without a court order.
- Exfiltration attacks are a major cybersecurity threat.



```
rekeying with psk_ke
static exfiltration of psk in T3:
```

✘	✘	✘	✘	✘	✘	✘	✘	...	✘	✘
T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	...	T_{n-1}	T_n

```
rekeying with key_update
static exfiltration of application_traffic_secret in T3:
```

✓	✓	✓	✘	✘	✘	✘	✘	...	✘	✘
T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	...	T_{n-1}	T_n

```
rekeying with (ec)dhe
static exfiltration of all keys in T3:
```

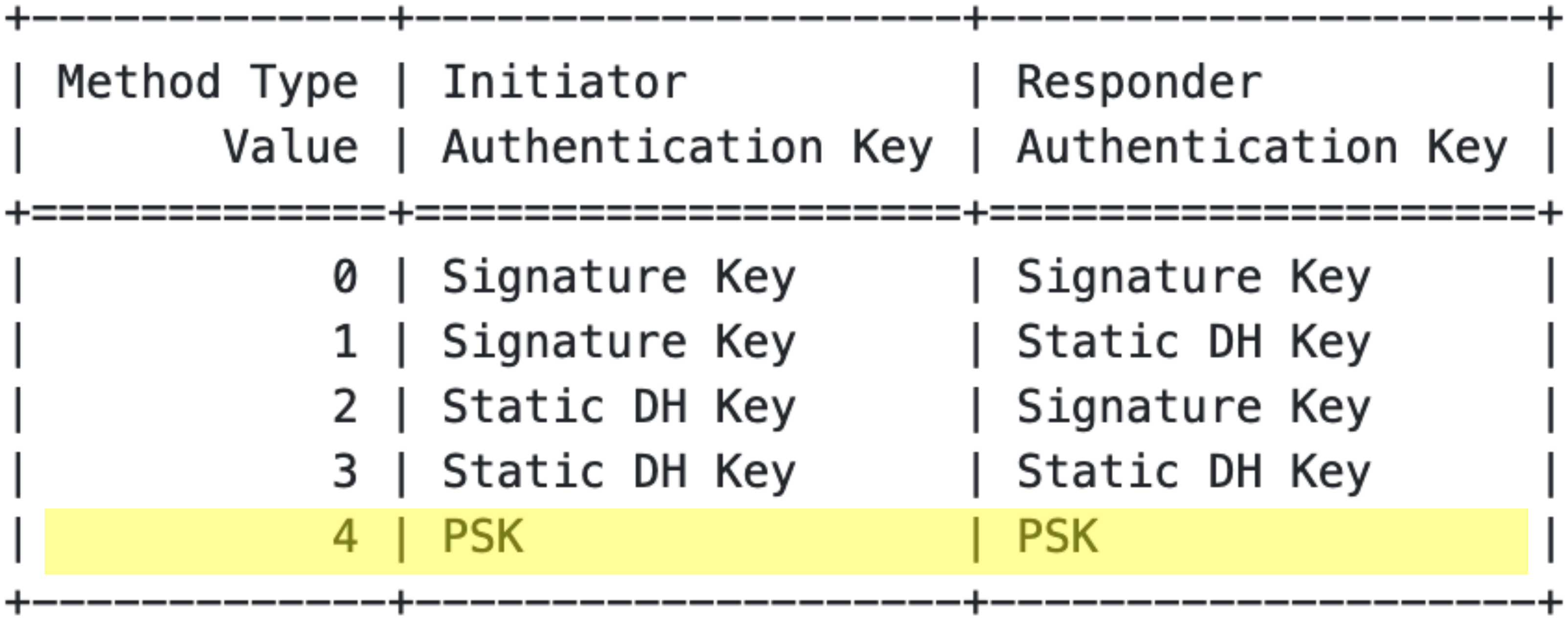
✓	✓	✓	✘	✓	✓	✓	✓	...	✓	✓
T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	...	T_{n-1}	T_n

Figure 1: Impact of static key exfiltration in time period T_3

New PSK authentication and resumption method

- Most of the following suggestions and discussion is takes from the LAKE mailing list https://mailarchive.ietf.org/arch/msg/lake/IeQJzeYqiaxB9sz30sXuayQZJs8/

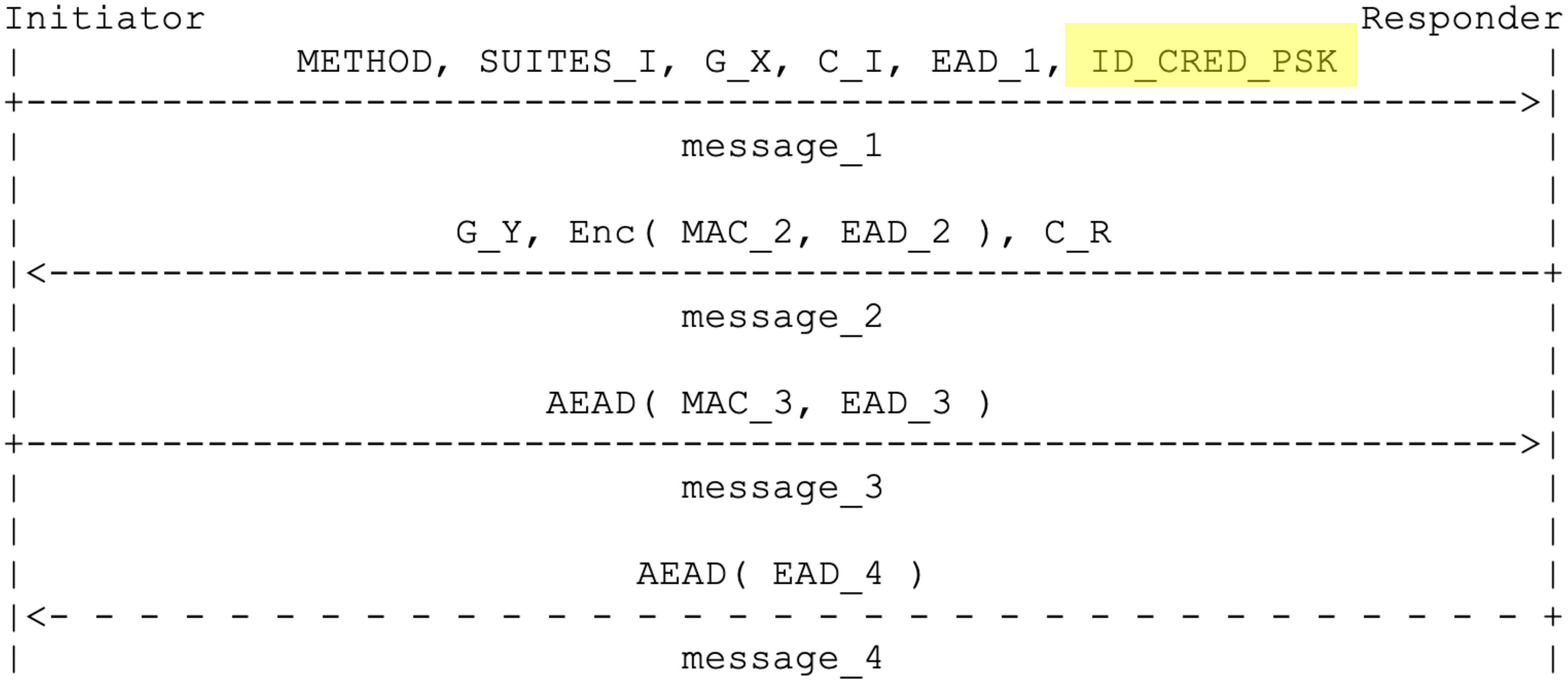
- **Benefits:**
  - Requires only one asymmetric operation compared to three in the current methods.
  - Eliminates external things like fetching credentials from a database, revocation, and path validation



Method Type Value	Initiator Authentication Key	Responder Authentication Key
0	Signature Key	Signature Key
1	Signature Key	Static DH Key
2	Static DH Key	Signature Key
3	Static DH Key	Static DH Key
4	PSK	PSK

New PSK authentication and resumption method

- Add ID_CRED_PSK in message_1. Remove ID_CRED_R in message_2 and ID_CRED_I in message_3
- Add PSK to the salt used to derive PRK_2e, e.g., salt = [TH_2, PSK]
- Derive PRK_3e2m ≠ PRK_4e3m ≠ PRK_2e
  - Suggested by Charlie Jacomme to prevent MAC oracle
- Derive resumption PSK = EDHOC_KDF(PRK_out, 11, h'', hash_length)
  - Only needed if initial method is 0-4. Then used only once. PSKs can be reused.



```
Initiator                                                       Responder
|            METHOD, SUITES_I, G_X, C_I, EAD_1, ID_CRED_PSK            |
+--------------------------------------------------------------------->|
|                              message_1                               |
|                                                                      |
|                 G_Y, Enc( MAC_2, EAD_2 ), C_R                        |
|<---------------------------------------------------------------------+
|                              message_2                               |
|                                                                      |
|                     AEAD( MAC_3, EAD_3 )                             |
+--------------------------------------------------------------------->|
|                              message_3                               |
|                                                                      |
|                         AEAD( EAD_4 )                                |
|<- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
|                              message_4                               |
```

Discussion

- Should message_1 contain a MAC or an AEAD? An AEAD enables encryption of C_I and EAD_1.
  A MAC/AEAD stops an attacker to create new message_1 but an attacker can replay an old message_1.
- How to provide identity protecting, mitigate identification, and mitigate tracking?
  Reusing a PSK identifier several times without it being encrypted enables identification and tracking.
  Several ways to achieve acceptable privacy with symmetric crypto. What to do if things get out of sync?
  - Require external PSK identifiers to be "privacy-friendly". This is mandated in all recent EAP methods.
  - Derive a new random PSK identifier in each EDHOC exchange. 64-bit would be enough.
    E.g., PSK ID = EDHOC_KDF(PRK_out, 12, h'', 8)
  - If message_1 contains a MAC/AEAD the Responder can test several PSK. The PSK Identifier can then be much smaller. Maybe 16, 24 or 32 bits.
  - The server sends a new PSK ID in message_2. This PSK ID can be much smaller, and the length can be chosen by the Responder. The disadvantage is that this increases the size of message_2, but message_2 could still be 45 bytes if there is no ID_CRED_R.
  - If message_1 contains a MAC (or AEAD) the Responder can test all PSKs. Symmetric operations are much cheaper than asymmetric. Typically, 2-3 orders of magnitude. So, testing 50 MACs is much cheaper than verifying a signature.

Discussion

- Combinations of PSK and Asymmetric authentication are possible but probably not worth doing. https://mailarchive.ietf.org/arch/msg/lake/_S6oMfv51k9R0OfcZrj_HF3hV1I/
- Charlie: Should we make sure that a PSK derived from some EDHOC exchange with some cipher suite SUITE can only be used for resumption with the corresponding SUITE or higher?
  - TLS 1.3 binds a PSK to a hash function to avoid the same PSK being used in several MAC functions. TLS does not restrict the cipher suite. Needs to be evaluated.
- Charlie: We need to be careful with selfie attacks.
  - This was discussed earlier when EDHOC had an PSK method. Several solutions exist.
- Charlie: if the PSK was derived in a post quantum secure way, even a diffie helman resumption would probably be post quantum secure, similar to IKE (https://datatracker.ietf.org/doc/html/rfc8784).
  - Yes, very likely. Only apply to external PSK. Resumption PSKs are not quantum-resistant.
- Rafa: Should we also have a 2-message resumption method similar to TLS 1.3 key update.
  - Does not provide aliveness and therefore not peer awareness either.
  - A 2-message resumption could add random numbers or ECDHE that TLS key update does not have.
- Rafa: Is ID_CRED_PSK protection a MUST? In the case of resumption, you could periodically run a full EDHOC authentication and reuse the PSK identity in the meantime.
  - Reuse could maybe be allowed for a short time, same location, same access.